MLT-04 Summary Document

Overview

This lecture gives a framework for developing a trading strategy using machine learning. The entire workflow is covered- from importing data, preprocessing it, feature engineering and then creating and evaluating a model.

The lecture covers the following topics:

- Importing data
- Data Resampling
- Feature Engineering
- Feature selection and parameter optimization
- Train-test-validation split
- Ensemble models to predict the sign of the future return
- Ensemble models using dollar bars
- Ensemble models combining features computed using two different time frames

Importing data

Data is imported from a trusted source. In the lecture, SPY data is imported from TradeStation into a pandas DataFrame.

Data Resampling

The data needs to be aggregated into larger timeframes. It is resampled into 5m, 30m, 1h, 1d, and 1w.

Feature engineering

This is done in three steps:

Define the functions for the features:

Functions to create the relevant features (columns) like log returns, internal bar strength, aqr momentum, vwap, rolling Bollinger bands, rolling acceleration (the second derivative of price), rolling volatility, RSI, rolling autocorrelation, etc. are computed along with smoothing functions on price like average price and rolling max and min.

Creating data store for features

Since the above features will be computed for the data resampled for different intervals, it is better to save it.

Create the features

Create the features using the functions defined in step (i) using pipelines, according to frequency and periods and save to hdf store.

Now we evaluate the features that were computed to see which features (if any) had predictive value intraday using the SPY ETF.

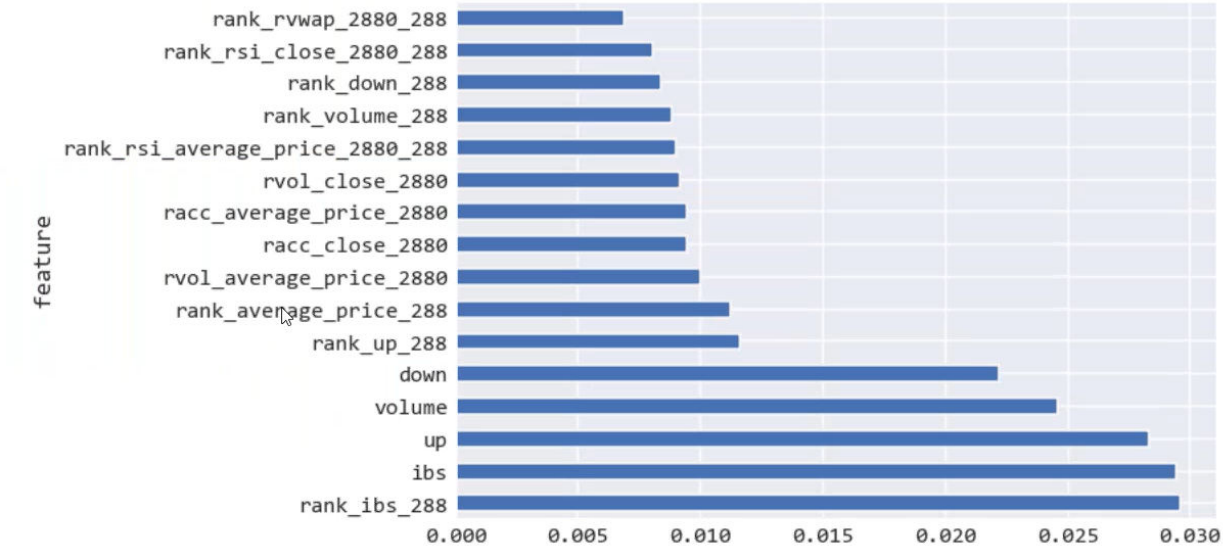
Feature selection and parameter optimization

We run Shapley Feature selection and select the shap features that exceed the mean value. These are the features that have the most impact on our data and remove the noisy features. The SHAP metrics are used because of their efficiency and consistency.

```python
def shap_imp(clf, X):
    explainer = shap.TreeExplainer(clf)
    shap_values = explainer.shap_values(X)

    fi0 = np.abs(shap_values[0]).mean(axis=0)
    fi1 = np.abs(shap_values[1]).mean(axis=0)
    fi = fi0 + fi1
    imp = pd.DataFrame({"feature": X.columns.tolist(), "mean": fi})
    imp = imp.set_index("feature")

    return imp
```



Now we will optimize the model itself by doing hyperparameter optimization. Neptune.ai is a free tool to optimize the model hyperparameters using the training set and validation set.

Train-validation-test split

The data is split into the training, validation, and test datasets by slicing the DataFrame. The test data should be left out as the holdout data till the very end to avoid overfitting. Therefore, all the optimizations should be done on the train and validate datasets.

Ensemble models to predict the sign of the future return

We will try to predict the sign of the 30m-ahead return using three ensemble models. We will compare the performance of the three models by plotting the multiclass ROC curve, multiclass precision-recall curve, and the classification matrix for each of them. We also compare the models based on different metrics like accuracy, precision, recall, Matthews Correlation Coefficient, specificity, f1 score, and Cohen's kappa coefficient.

A Random forest model:

It is imported from sklearn.ensemble library and initialized as shown below.

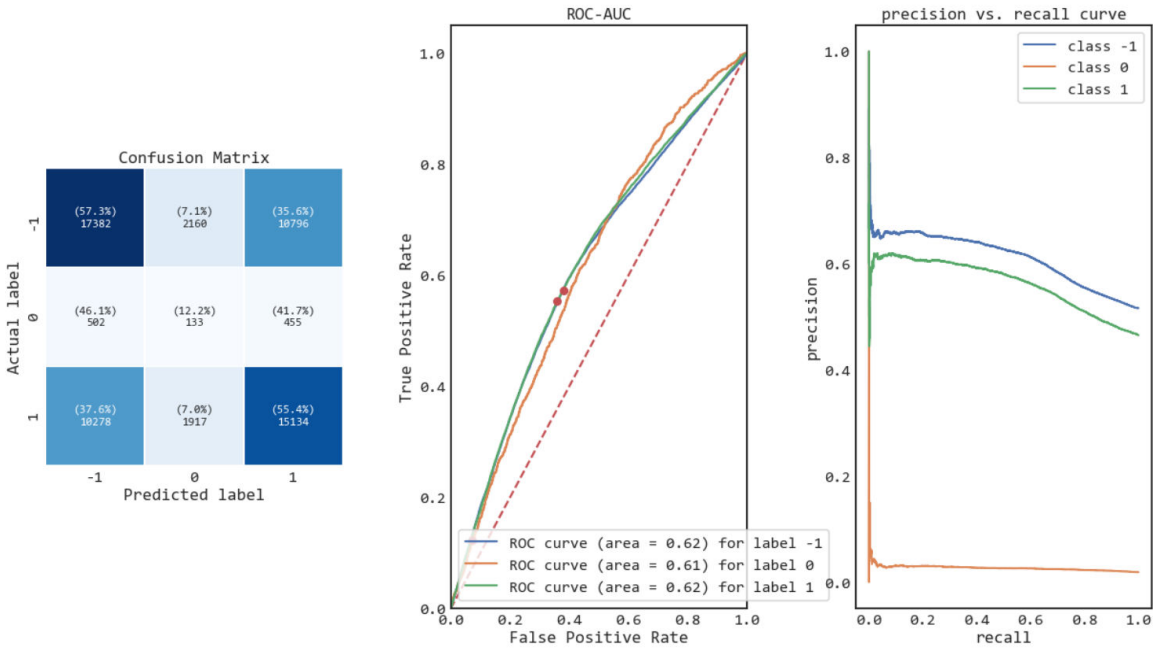
```
rf_clf_ = RandomForestClassifier(
    criterion="entropy",
    max_depth=7,
    class_weight="balanced_subsample",
    n_estimators=2000,
    random_state=RANDOM_STATE,
    oob_score=True,
    n_jobs=-1,
)
```

The model is fitted on the training data set, initialised and its performance is evaluated on the test data set.

```
rf_clf_.fit(X_train_, y_train_)

rf_clf_perf = performance_evaluation_report_multiclass(
    rf_clf_,
    X_test_,
    y_test_,
    show_plot=True,
    show_pr_curve=True,
    average=None,
    labels=classes,
)
```

This model's non-optimized performance is the best out of our three models.



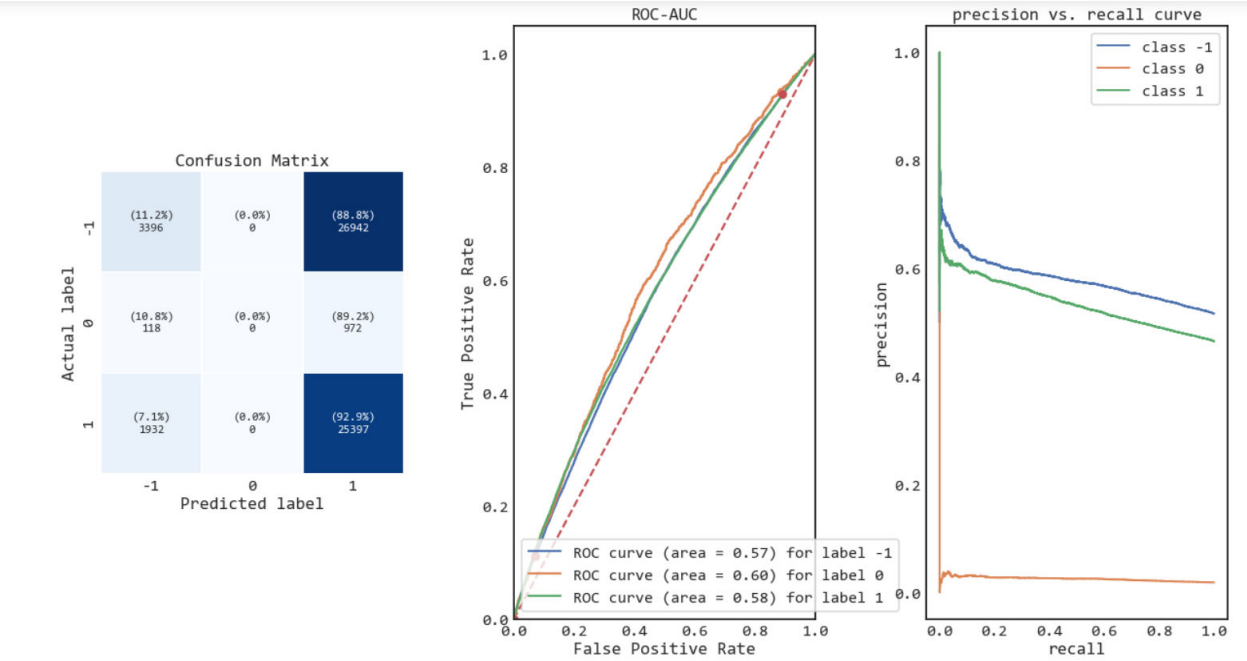
We see the performance of this model improve further when we run it using the top shap features.

A Bagging classifier model

It is initialized as shown below. First, we create a Decision Tree classifier (from sklearn.tree) and then pass it to the bagging classifier (from sklearn.ensemble) along with the other parameters.

```
clf_base = DecisionTreeClassifier(
    criterion="entropy",
    max_features=1,
    class_weight="balanced",
    min_weight_fraction_leaf=0,
    random_state=RANDOM_STATE,
)
bag_clf = BaggingClassifier(
    base_estimator=clf_base,
    n_estimators=2000,
    max_features=0.05,
    max_samples=0.05,
    oob_score=True,
    n_jobs=-1,
    random_state=RANDOM_STATE,
)
```

The model is fitted on the training data set, and its performance is evaluated on the test data set. This model did not do well in predicting the sign of the returns because it was always predicting low.

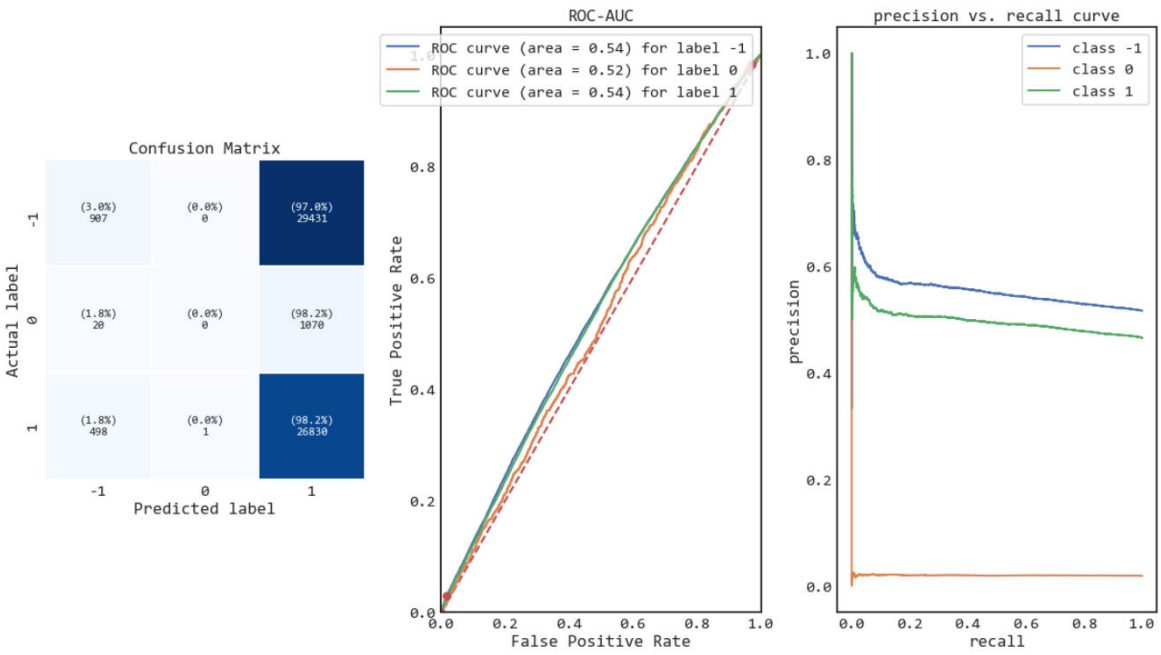


A LightGBM model

The lightgbm library is imported, and the model is initialised as shown below.

```
gbm = lgb.LGBMClassifier(
    boosting_type="dart",
    learning_rate=0.15,
    feature_fraction=0.7,
    bagging_fraction=0.7,
    n_estimators=2000,
    silent=True,
    n_jobs=-1,
    random_state=78,
    # predict_disable_shape_check=True,
)
```

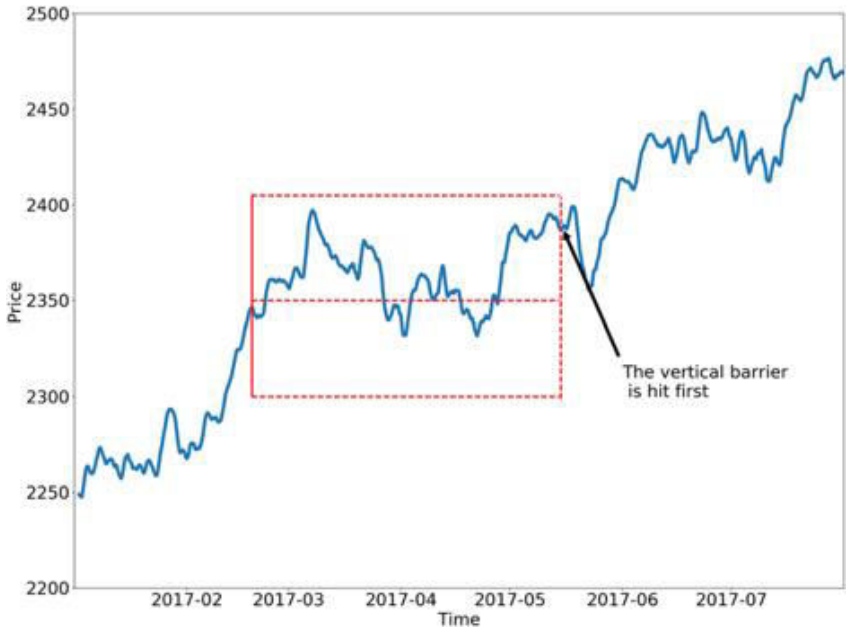
The model is fitted on the training data set, and its performance is evaluated on the test data set. This model also did not do very well.



Ensemble models using dollar bars

We will use the dollar bars to reduce the size of the data set and improve the features' predictive power. Dollar bars are simply price bars created by sampling data after an equal amount of dollar volume is traded. In this experiment, that value is set to 10 million USD.

We also implement the triple barrier labelling method, which incorporates asset volatility in setting the profit targets or stop-losses for our labels. It also incorporates a time barrier which outputs a label zero if neither the profit target nor stop loss are hit before the time is up. For more information on triple barrier labelling, optimize, and hyperparameters, see here.



Source

The Random Forest, Bagging Classifier, and LightGBM are against fitted on this data and their performances evaluated. LightGBM is the only model that predicted the positive class at all.

We again optimize the features and create the models using the top shap features. We see that the Random Forest Classifier outperforms again.

Now we tune the hyperparameters using neptune.ai and run the models on the holdout dataset with the best hyperparameters.

Ensemble models combining features computed using two different time frames

Now we use the same mlfinlab process (as in point 6 above) after incorporating features from the 30-minute time frame along with the 5 min data. We compute the dollar bars, split the data into train, validate, and test datasets. Then we run the three ensemble models first with the unoptimized parameters, then with the top shap features, and finally with the optimized hyperparameters from neptune.ai. We analyze the performance of the three models similarly.